Phát hiện mã độc trên Android bằng phương pháp phân tích tập tin manifest

Lê Bá Cường, Trịnh Doãn Mạnh

***Tóm tắt*— Mã độc trên điện thoại chạy hệ điều hành Android ngày càng nhiều. Do vậy, việc phân tích các ứng dụng trước khi thực hiện cài đặt lên thiết bị là rất cần thiết. Trong các phương pháp phân tích thì phân tích tĩnh là phương pháp cho kết quả khá chính xác và tiết kiệm nhất. Bài báo trình bày phương pháp phát hiện mã độc trên thiết bị di động bằng cách phân tích tĩnh các thuộc tính thu được từ tệp tin manifest của ứng dụng. Phương pháp này có thể được sử dụng để phát hiện các mẫu mã độc không bị phát hiện bằng phương pháp phân tích dựa trên chữ ký. Phương pháp phân tích của chúng tôi gồm bốn bước sẽ được trình bày chi tiết trong bài báo này. Sau các bước phân tích sẽ đưa ra kết luận ứng dụng đưa vào kiểm tra có an toàn hay không. Qua đó, giúp người dùng tránh được việc cài các ứng dụng độc hại lên thiết bị của mình.**

***Abstract*— Malicious code on phones is running on Android operating system and more. Therefore, the analysis of the application before installating on the device that is very necessary. In the analytical methods, static analysis is the method that gives the most accurate and economical results. The article presents the method of detecting malicious code on mobile phones using attributes obtained from the application's manifest file. This method can be used to detect malware samples that are not detected by signature-based analysis. Our four-step analysis method will be covered in detail in this article. After the analysis steps, it will be concluded whether the application included in the test is safe or not. This helps users avoid installing malicious applications on their devices.**

***Từ khóa*— Mã độc trên Android; phân tích động; phân tích tĩnh; tệp tin manifest; quyền nguy hiểm trên Android.**

***Keywords*— Android malware; dynamic analysis; static analysis ; manifest file; dangerous permissions on Android.**

Bài báo được nhận ngày 3/6/2017. Bài báo được gửi cho phản biện thứ nhất vào ngày 17/7/2017 và nhận được ý kiến đồng ý đăng của phản biện thứ nhất vào ngày 1/8/2017. Bài báo được gửi cho phản biện thứ hai vào ngày 24/7/2017 và nhận được ý kiến đồng ý đăng của phản biện thứ hai vào ngày 1/8/2017.

I. GIỚI THIỆU

Điện thoại thông minh đã trở nên rất phổ biến trong vài năm trở lại đây. Trong sự phát triển của thị trường di động thông minh, Android-một nền tảng mã nguồn mở của Google đã trở thành một trong những hệ điều hành di động phổ biến nhất, chủ yếu được sử dụng trên điện thoại thông minh và máy tính bảng, được trang bị một số tính năng như kết nối wifi, gọi điện, lưu trữ dữ liệu, định vị toàn cầu (GPS),....

Đi đôi với sự phát triển của hệ điều hành Android thì số lượng mã độc phát triển trên hệ điều hành này cũng ngày một tăng cao. Năm 2012, số lượng mã độc phát hiện mới trên nền tảng Android là 214.327, đến năm 2016 đã tăng lên 3.246.284 mẫu mã độc được phát hiện mới [1]. Điều này dẫn đến các phần mềm phòng chống mã độc trên Android cần phải cải tiến về phương pháp và kỹ thuật. Đã có rất nhiều nghiên cứu tập trung vào việc phát hiện phần mềm độc hại trên Android. Một trong những phương pháp phổ biến bao gồm các phương pháp dựa trên chữ ký, trích chữ ký từ phần mềm độc hại mẫu. Isohara trong [16] đã trình bày một phương pháp phát hiện phần mềm độc hại bằng cách phân tích các thuộc tính của các tập tin trong các mẫu ứng dụng. Mặc dù cách tiếp cận này có thể phát hiện một số phần mềm độc hại mà không bị phát hiện bởi blacklist hoặc phương pháp phân tích dựa trên chữ ký, chi phí phân tích phụ thuộc vào số lượng tệp trong mẫu phân tích. Enck và cộng sự [17] đề xuất một phương pháp ngăn chặn việc cài đặt các ứng dụng có quyền truy cập ở mức nguy hiểm hoặc intent filter (một cơ chế để thực hiện hợp tác giữa các ứng dụng Android). Tuy nhiên, phương pháp này có thể dẫn đến phát hiện không chính xác, bởi vì thông tin được sử dụng trong phương pháp không đủ để phân biệt phần mềm độc hại từ các ứng dụng lành tính. Ngoài ra, còn phương pháp phân tích mã độc dựa trên việc phân tích các lời gọi API trong tệp tin smali như trong nghiên cứu của Wu và cộng sự [18]. Tuy nhiên, việc thực hiện phương pháp trên sẽ nảy sinh vấn đề là chi phí phân tích rất lớn, nó tùy thuộc vào số lượng tệp tin và kích thước của tệp tin trong ứng dụng ban đầu.

Qua kết quả phân tích kích thước và số lượng tệp tin từ 20 mẫu mã độc và 20 ứng dụng lành tính, chúng tôi tổng hợp lại qua 2 bảng sau:

BẢNG 1. BẢNG KÍCH THƯỚC TRUNG BÌNH CÁC TẬP TIN (ĐƠN VỊ: KB)

	Tệp tin smali	**Tệp tin mã nguồn**	**Tệp tin manifest**
20 mẫu mã độc	4503	3670	5
20 mẫu lành tính	2614	1407	4

BẢNG 2. BẢNG SỐ LƯỢNG TRUNG BÌNH CÁC TẬP TIN

	Tệp tin smali	**Tệp tin mã nguồn**	**Tệp tin manifest**
20 mẫu mã độc	583	342	1
20 mẫu lành tính	256	102	1

Tệp tin manifest là tệp bắt buộc phải có trong bất kì ứng dụng nào, dù là ứng dụng lành tính hay chứa mã độc. Số lượng tệp tin manifest trong mỗi ứng dụng chỉ có 1 tệp tin duy nhất, đồng thời kích thước của tệp tin manifest rất nhỏ. Vì vậy, trong bài báo này, chúng tôi nghiên cứu phương pháp phát hiện mã độc trên Android dựa vào các đặc trưng của tệp tin manifest. Phương pháp này sử dụng các kỹ thuật dịch ngược để trích xuất thông tin từ tệp tin manifest của ứng dụng. Với kỹ thuật này, người phân tích có thể tìm ra các ứng dụng độc hại mà không phải chạy ứng dụng trên thiết bị hoặc môi trường thử nghiệm. Điều này giúp rút ngắn thời gian phát hiện ra các ứng dụng độc hại. Đồng thời, kỹ thuật phân tích này cũng có tỉ lệ phát hiện cao và tiêu tốn ít tài nguyên.

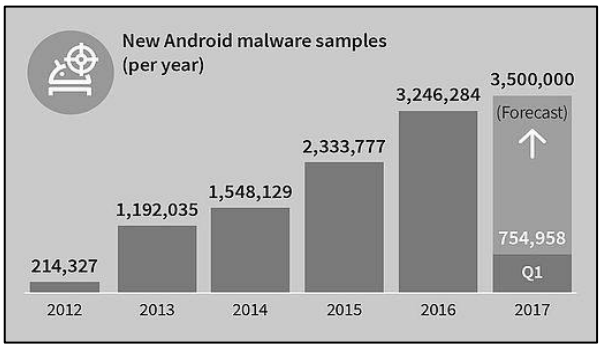
Phần còn lại của bài báo được bố cục như sau: Mục II đưa ra các vấn đề về mã độc trên hệ điều hành Android; Mục III trình bày các kỹ thuật phân tích mã độc trên Android; Mục IV trình bày mô hình và các bước triển khai chi tiết của phương pháp; Kết quả thực nghiệm, đánh giá được trình bày trong mục V và cuối cùng là mục Kết luận, trong đó trình bày định hướng nghiên cứu và phát triển của nhóm tác giả.

II. MÃ ĐỘC TRÊN HỆ ĐIỀU HÀNH ANDROID

Điện thoại di động đã trở thành một mục tiêu của các loại tội phạm mạng. Điện thoại di động có chứa một lượng lớn các dữ liệu cá nhân như danh bạ, email, đồng thời cũng có thể giúp người dùng thực hiện các giao dịch trực tuyến và các tiện ích khác.

Một mã độc bất kỳ có thể là các mã độc hại hoặc các phần mềm độc hại được thiết kế để thực hiện cả các chức năng mà không có sự đồng ý của người dùng.

Tồn tại nhiều phần mềm độc hại cố gắng đánh cắp dữ liệu hoặc dịch vụ trên điện thoại. Các phần mềm gián điệp có khả năng truy cập vào micro, máy ảnh và dịch vụ định vị toàn cầu GPS và gửi dữ liệu đó về cho kẻ tấn công. Phần mềm quảng cáo là loại phần mềm độc hại mà sử dụng các kênh truyền thông hiện có như: Email, MMS, SMS hay Bluetooth để truyền những tin quảng cáo không mong muốn tới số điện thoại hay người dùng có trong danh bạ của máy nạn nhân.



Hình 1. Số lượng mã độc phát hiện mới theo từng năm

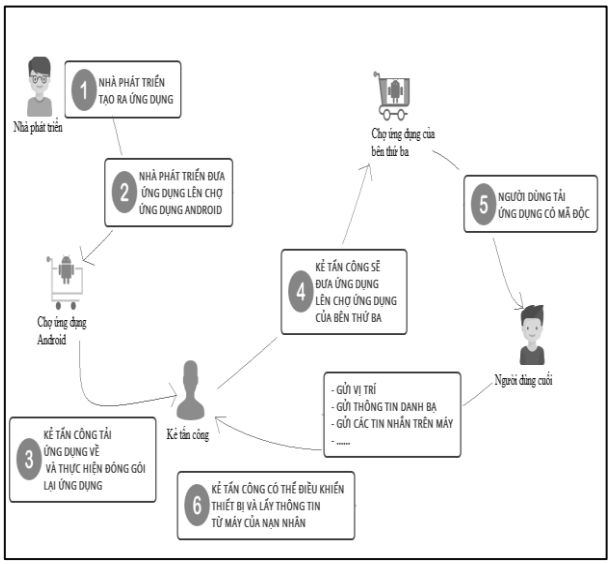
Theo thống kê của GDATA (Google Data) (Hình 1), số lượng mã độc ngày càng tăng nhanh. Có ba yếu tố chính dẫn tới việc tăng nhanh về số lượng các mã độc trên nền tảng di động Android đó là: hệ điều hành mở, hệ điều hành di động phổ biến nhất và động cơ của các tội phạm mạng [2].

Android là một hệ điều hành di động mã nguồn mở. Ngoài ra, Android cho phép người dùng phát triển ứng dụng Android và các ứng dụng đó có thể được xuất bản mà không cần xem xét bởi bên thứ ba [3]. Chính vì vậy mà các kẻ tấn công đã tận dụng cơ hội này để phát triển phần mềm độc hại tấn công lên hệ điều hành Android. Tuy nhiên, ứng với mỗi phiên bản hệ điều hành Android khác nhau, chạy trên các thiết bị khác nhau sẽ có những lỗi khác nhau. Các lỗi còn phụ thuộc vào việc tùy chỉnh các khả năng bảo mật của các nhà sản xuất thiết bị [3]. Vì vậy, các kẻ tấn công sẽ lựa chọn hệ điều hành Android làm mục tiêu nếu hệ điều hành này được nhiều người sử dụng.

Google Play là chợ ứng dụng chính thức của Android, nơi cung cấp hàng ngàn ứng dụng miễn phí hoặc trả tiền. Các ứng dụng này bao gồm: trò chơi, mạng xã hội, đa phương tiện,.... Tuy nhiên, Android là một hệ điều hành mở, đồng thời người dùng có thể cài đặt ứng dụng Android từ nhiều nguồn khác ngoài Google Play [4]. Vì lý do đó,

tin tặc có thể dễ dàng tạo ra các ứng dụng độc hại và phân phối nó trực tiếp tới người dùng một cách dễ dàng.

Một ứng dụng độc hại có thể được tạo ra dễ dàng bằng cách chèn các đoạn mã độc vào các ứng dụng lành tính ban đầu. Như trong Hình 2 ta có thể thấy rõ quy trình đó. Đó là lý do trong bài báo này trình bày về kỹ thuật phân tích tĩnh.



Hình 2. Quá trình đóng gói lại một ứng dụng

Ngoài các lý do trên thì trách nhiệm từ phía người dùng là không nhỏ. Do người dùng thiếu kiến thức về mã độc nên dễ dàng chấp nhận cài đặt các ứng dụng từ phía các nhà cung cấp không tin cậy. Vì vậy, việc phân tích ứng dụng trước khi thực hiện cài đặt là một biện pháp để hỗ trợ người dùng giúp họ tránh được các ứng dụng chứa mã độc.

III. CÁC KỸ THUẬT PHÂN TÍCH MÃ ĐỘC TRÊN ANDROID

Hai kỹ thuật chủ đạo đó gồm, kỹ thuật phân tích động và kỹ thuật phân tích tĩnh.

Kỹ thuật phân tích động còn được gọi là phân tích hành vi, được sử dụng để phân tích và nghiên cứu hành vi của phần mềm độc hại. Sau đó nghiên cứu các cách phần mềm độc hại tương tác với hệ thống, dịch vụ, thu thập dữ liệu, thực hiện kết nối mạng, mở cổng dịch vụ,... Trong giai đoạn này, các tập tin *apk* được cài đặt trên một thiết bị mô phỏng (hoặc thiết bị thật) để quan sát hành vi của ứng dụng. Ví dụ, các nghiên cứu về kỹ thuật phân tích động như trong [7-9].

Trong nghiên cứu [7], quyền định nghĩa trong tệp tin manifest của ứng dụng Android sẽ được chia làm 2 phần: Quyền tiêu chuẩn và quyền không theo tiêu chuẩn. Sau đó, biểu đồ Control Flow Graph (CFG) sẽ được sử dụng để phát hiện mã độc.

Trong nghiên cứu [8], một số ứng dụng nghi ngờ có Server Command and Control (C&C) sẽ được tải xuống và phân tích. Sau đó một đồ thị quan hệ sử dụng Gephi để xây dựng. Các API nhạy cảm và các Intent-filter sẽ được tìm kiếm trong đồ thị. Thuật toán A* được sử dụng để tìm chi phí đường đi thấp nhất trong ứng dụng lành tính và độc hại từ đồ thị.

Trong nghiên cứu [9], các lời gọi hàm hệ thống trong ứng dụng được sử dụng để phát hiện mã độc. Các lời gọi hàm hệ thống của ứng dụng được giám sát trên layer Android architecture.

Tuy nhiên, kỹ thuật phân tích động thường cần rất nhiều chi phí để phân tích, do đó thường không thể đưa được ra ngay kết quả phân tích trong thời gian ngắn.

Kỹ thuật phân tích tĩnh sẽ sử dụng các kỹ thuật dịch ngược để trích xuất thông tin từ mã nguồn của ứng dụng. Với kỹ thuật này, người ta có thể tìm ra các ứng dụng độc hại mà không phải chạy ứng dụng trên thiết bị hoặc môi trường thử nghiệm [5]. Điều này giúp rút ngắn thời gian trong quá trình phát hiện ra các ứng dụng độc hại. Thêm vào đó, kỹ thuật phân tích tĩnh cũng có tỉ lệ phát hiện cao hơn và tiêu tốn ít tài nguyên hơn [5]. Các nghiên cứu về kỹ thuật phân tích tĩnh có trong [5] và [12].

Trong nghiên cứu [5], Intent trong ứng dụng Android được sử dụng để phân loại ứng dụng thuộc loại lành tính hoặc phần mềm độc hại. Các Intent được thực hiện bằng cách trích xuất từ tệp *apk* của ứng dụng Android. Sử dụng APKTool để lấy *byte code*. Sau đó được Intent sử dụng để gắn thẻ tính năng. Tiếp đó, một đoạn mã C++ sẽ được sử dụng để phân loại. Tính năng của các *tag* được phân loại và việc phân loại được sử dụng để chỉ ra các tính năng của các ứng dụng độc hại.

Trong nghiên cứu [12], sử dụng kỹ thuật phân tích tĩnh để phát hiện các ứng dụng lành tính và độc hại bằng cách sử dụng công cụ dịch ngược Androguard để lấy ra các API được sử dụng trong ứng dụng đó. Sau đó, sử dụng DroidAPIMiner để phát hiện ứng dụng lành tính hay không.

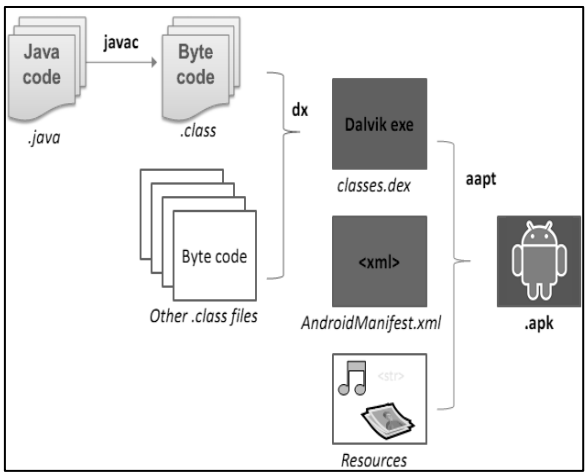
Phát hiện ứng dụng độc hại trên Android có thể được phân tích bằng cách sử dụng kỹ thuật phân tích tĩnh hoặc kỹ thuật phân tích động, đôi khi cả hai kỹ thuật được sử dụng đồng thời.

Trong bài báo này, chúng tôi đề xuất phương pháp để phát hiện phần mềm độc hại Android bằng

cách phân tích tệp tin manifest. Mỗi ứng dụng Android phải có tệp tin manifest, trong đó trình bày thông tin thiết yếu về ứng dụng. Phương pháp đề xuất của chúng tôi dựa trên phân tích đặc trưng của tệp tin Android manifest và có hiệu quả để phát hiện các phần mềm độc hại đã được xác định và phần mềm độc hại chưa được xác định. Hơn nữa, chi phí cho phương pháp này khá thấp, bởi vì phương pháp này chỉ phân tích tệp tin manifest. Tệp tin manifest có kích thước rất nhỏ so với kích thước của các tệp tin như Resource hay smali.

IV. MÔ HÌNH VÀ CÁC BƯỚC TRIỂN KHAI CỦA PHƯƠNG PHÁP

Một ứng dụng Android bao gồm các phần như sau:



Hình 3. Các thành phần của tệp tin ứng dụng

AndroidManifest.xml: tập cung cấp thông tin cần thiết để ứng dụng hoạt động bình thường với hệ thống Android, trong đó hệ thống sẽ đọc tệp tin này trước khi có thể chạy các đoạn mã khác của ứng dụng. Trong đó có một số thông tin như: các quyền mà ứng dụng yêu cầu, các API tối thiểu để ứng dụng hoạt động, danh sách thư viện ứng dụng cần,...

Classes.dex: mã nguồn Java được biên dịch để chạy trong máy ảo Dalvik.

Resources: bao gồm hai phần chính, một thư mục *res* chứa các tài nguyên cần thiết cho ứng dụng không được biên dịch trước như: ảnh, *String*,... và tệp tin *resources.arsc* chứa tài nguyên đã được biên dịch trước.

META-INF: thư mục này chứa một số metadata như chứng chỉ số của ứng dụng, tệp tin manifest của ứng dụng java.

Lib: các thư viện được biên dịch sẵn phù hợp với từng nền tảng phần cứng.

Assets: chứa các tài nguyên mà ứng dụng có thể truy cập thông qua AssetManager.

Mô hình phương pháp để phát hiện phần mềm độc hại Android bằng cách phân tích tệp tin manifest được thể hiện như Hình 4.

Phần mềm độc hại được phát hiện theo các bước sau:

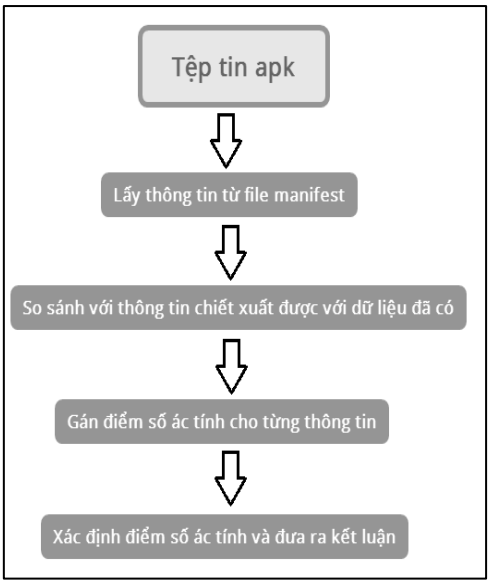
Bước 1: Trích xuất thông tin cụ thể trong tệp tin manifest.

Bước 2: So sánh thông tin trích xuất được với dữ liệu đã có.

Bước 3: Sau đó tính điểm số ác tính cho các thông tin trích xuất được.

Bước 4: Dựa vào điểm số ác tính để phân loại ứng dụng: là lành tính hay chứa mã độc.

Phương pháp phân tích sẽ như sau:



Hình 4. Sơ đồ phương pháp phân tích

A. Trích xuất thông tin

Tệp tin manifest có các thông tin về các ứng dụng Android, chẳng hạn như phiên bản của ứng dụng, tên của package, các permission phải có, và level API. Định dạng của tệp tin manifest giống hệt nhau trong cả ứng dụng lành tính và phần mềm độc hại. Tuy nhiên, ở đây có sự khác biệt nhất định trong các đặc tính của một số mục thông tin. Bằng thực nghiệm cũng như đọc các tài liệu liên quan. Với kết quả thực hiện điều tra trên 30 mẫu lành tính và 30 mẫu phần mềm độc hại, tổng cộng là 60 mẫu [13]. Trong bài báo này đã lựa chọn cụ thể ra một mục thông tin hiển thị trên nhiều loại phần mềm độc hại khác biệt so với các ứng dụng lành tính.

Dưới đây là một số thông tin được trích xuất từ tệp tin manifest để sử dụng trong phương pháp này bao gồm:

(1) Permission

(2) Intent filter (action)

(3) Intent filter (category)

(4) Process name

B. So sánh thông tin trích xuất với dữ liệu đã có

Trong phương pháp này, một số danh sách các từ khóa (keyword) được biên soạn cho một ứng dụng. Một ứng dụng là lành tính hay là ứng dụng độc hại thì chuỗi trong tệp tin manifest đều được ghi lại trong một danh sách với các từ khóa tương ứng. Ở đây sẽ có 4 kiểu danh sách từ khóa [13] bao gồm:(1) Permission, (2) Intent filter (action), (3) Intent filter (category), và (4) Process name.

Dưới đây là danh sách các Permission trong tệp tin manifest của các mẫu mà chúng tôi đã phân tích. Trong mỗi bảng, cột tên *Permission* là tên của quyền đó xuất hiện trong tệp tin manifest. Cột số lần xuất hiện ghi tổng số lần xuất hiện của quyền đó trong tất cả các mẫu được phân tích.

Đối với 40 mẫu lành tính, danh sách các quyền yêu cầu được trình bày trong Bảng 3.

BẢNG 3. DANH SÁCH CÁC QUYỀN YÊU CẦU TRONG ỨNG DỤNG LÀNH TÍNH

STT	Tên Permission	Số lần xuất hiện
1	INTERNET	20
2	WRITE_EXTERNAL_STORAGE	13
3	ACCESS_NETWORK_STATE	11
4	WAKE_LOCK	9
5	READ_PHONE_STATE	7
6	VIBRATE	9
7	ACCESS_WIFI_STATE	5
8	USE_CREDENTIALS	5
9	MANAGE_ACCOUNTS	4
10	SET_WALLPAPER	4

Đối với 40 mẫu mã độc, danh sách các quyền yêu cầu được trình bày trong Bảng 4.

BẢNG 4. DANH SÁCH CÁC QUYỀN YÊU CẦU TRONG MÃ ĐỘC

STT	Tên Permission	Số lần xuất hiện
1	READ_PHONE_STATE	35
2	INTERNET	32
3	SEND_SMS	28
4	WRITE_EXTERNAL_STORAGE	26
5	ACCESS_NETWORK_STATE	25
6	RECEIVE_SMS	25
7	READ_SMS	23
8	ACCESS_WIFI_STATE	19
9	WRITE_SMS	17
10	READ_CONTACTS	17

Dưới đây là một số thống kê các quyền được sử dụng của các dạng mã độc [14].

BẢNG 5. TOP 10 QUYỀN NGUY HIỂM MÀ CÁC SPY SỬ DỤNG

STT	Tên Permission	Số lần xuất hiện
1	READ_PHONE_STATE	845
2	SEND_SMS	802
3	INTERNET	799
4	RECEIVE_SMS	757
5	WRITE_EXTERNAL_STORAGE	732
6	READ_SMS	712
7	CALL_PHONE	701
8	READ_CONTACTS	501
9	CHANGE_WIFI_STATE	432
10	WRITE_SMS	387

BẢNG 6. TOP 10 QUYỀN NGUY HIỂM MÀ RANSOMEWARE SỬ DỤNG

STT	Tên Permission	Số lần xuất hiện
1	READ_PHONE_STATE	585
2	INTERNET	581
3	WRITE_EXTERNAL_STORAGE	201
4	GET_TASKS	165
5	CAMERA	128
6	RECEIVE_SMS	114
7	SYSTEM_ALERT_WINDOW	109
8	READ_CONTACTS	102
9	BATTERT_STATS	73
10	SEND_SMS	69

BẢNG 7. TOP 10 QUYỀN NGUY HIỂM MÀ TROJAN SỬ DỤNG

STT	Tên Permission	Số lần xuất hiện
1	INTERNET	3528
2	READ_PHONE_STATE	3245
3	SEND_SMS	2016
4	WRITE_EXTERNAL_STORAGE	1513
5	RECEIVE_SMS	1498
6	WRITE_SMS	1436
7	ACCESS_FINE_LOCATION	1247
8	ACCESS_COARSE_LOCATION	982
9	GET_TASKS	899
10	READ_SMS	631

Kết quả trên cho thấy xuất hiện một số quyền phổ biến. Các con số này cho thấy rằng, các quyền liên quan đến gửi tin nhắn (SMS), chẳng hạn như SEND_SMS, RECEIVE_SMS và READ_SMS thường được sử dụng bởi các mẫu mã độc. Các quyền này sẽ được cho vào danh sách các quyền nguy hiểm. Ta sẽ tiến hành tương tự để tìm ra các keyword phân loại cho (2) Intent filter (action), (3) Intent filter (category), và (4) Process name. Các danh sách keyword thu được sẽ được sử dụng trong quá trình đánh giá độ ác tính của các mẫu phân tích.

Kết quả thu thập từ các mẫu:

(Danh sách 1) Permission nguy hiểm bao gồm:

1. READ_SMS
2. SEND_SMS
3. RECEIVE_SMS
4. WRITE_SMS
5. PROCESS_OUTGOING_CALLS
6. MOUNT_UNMOUNT_FILESYSTEMS
7. READ_HISTORY_BOOKMARKS
8. WRITE_HISTORY_BOOKMARKS
9. READ_LOGS
10. INSTALL_PACKAGES
11. MODIFY_PHONE_STATE
12. READ_PHONE_STATE
13. INTERNET
14. CALL_PHONE

(Danh sách 2) Intent-filter (action) nguy hiểm bao gồm:

1. BOOT_COMPLETED
2. SMS_RECEIVED
3. CONNECTIVITY_CHANGE
4. USER_PRESENT
5. PHONE_STATE
6. NEW_OUTGOING_CALL
7. UNINSTALL_SHORTCUT
8. INSTALL_SHORTCUT
9. left_up
10. right_up
11. left_down
12. right_down
13. SIG_STR
14. VIEW (keyword lành tính)

(Danh sách 3) Intent-filter (category) nguy hiểm bao gồm:

1. HOME
2. BROWSABLE (keyword lành tính)

(Danh sách 4) Process name nguy hiểm:

1. remote2
2. main
3. two
4. three

Sau khi thực hiện thu thập được danh sách từ khóa, ta sẽ thực hiện tính điểm số ác tính cho các thông tin thu được từ tệp tin manifest. Đầu tiên, ta cần thực hiện phân loại các thuộc tính thu được từ tệp tin manifest là lành tính hay độc hại. Sau đó, áp dụng công thức sau để tính [13]:

$$P = \frac{M - B}{E} \quad (1)$$

Trong đó:

P là điểm số ác tính

M là số lượng chuỗi độc hại

B là số chuỗi lành tính

E là tổng số thông tin thu được

Ví dụ như sau:

Trong tệp tin manifest ta thu được các quyền như sau:

<uses-permission android:name="android.permission.READ_PHONE_STATE"/>

<uses-permission android:name="android.permission.READ_CONTACTS"/>

<uses-permission android:name="android.permission.WRITE_CONTACTS"/>

<uses-permission android:name="android.permission.INTERNET>

<uses-permission android:name="android.permission.CALL_PHONE"/>

<uses-permission android:name="android.permission.GET_TASKS"/>

<uses-permission android:name="android.permission.READ_LOGS"/>

Có tất cả là 7 quyền, trong đó các quyền INTERNET, CALL_PHONE, READ_LOGS và READ_PHONE_STATE có trong danh sách các quyền nguy hiểm.

Vì vậy, điểm số ác tính của mẫu được tính theo công thức (1) là:

$$P = \frac{4-0}{7} = 0.57$$

Thực hiện đánh giá tương tự với các thuộc tính còn lại gồm: (2) Intent filter (action), (3) Intent filter (category), và (4) Process name.

C. Đưa ra đánh giá

Phương pháp đề xuất cung cấp các giá trị ngưỡng cho điểm số ác tính. Để có giá trị ngưỡng chính xác ta cần áp dụng các kỹ thuật thống kê trung bình và trong phương pháp với tập dữ liệu đủ lớn để có được giá trị ngưỡng tốt nhất cho điểm số ác tính. Bằng cách sử dụng bốn thông tin thu được từ (1), (2), (3) và (4) dựa vào việc phân tích với 50 mẫu mã độc chúng tôi đề xuất giá trị ngưỡng là 0.52.

Kết luận cuối cùng dựa trên cơ sở của điều kiện 1, 2 và công thức được đưa dưới đây. Điều kiện 1 mô tả đặc tính của phần mềm độc hại. Điều kiện 2 được thực hiện để tránh các kết luận không chính xác. Trong công thức dưới đây, SCORE được đề cập là điểm số ác tính cuối cùng của mẫu. C1 và C2 đếm số lượng hài lòng của một mẫu ở điều kiện 1 và 2 tương ứng.

Điều kiện 1:

- Điểm số ác tính của Permission lớn hơn giá trị ngưỡng.
- Điểm số ác tính của Process Name lớn hơn giá trị ngưỡng.

Điều kiện 2:

- Điểm của Intent-filter (action) là số âm (có giá trị < 0)
- Điểm của Intent-filter (category) là số âm (có giá trị < 0)

Công thức:

$$SCORE = C1 - C2$$

Nếu *SCORE* cuối cùng lớn hơn hoặc bằng 1, thì mẫu thử nghiệm được coi là phần mềm độc hại.

V. KẾT QUẢ THỰC NGHIỆM VÀ ĐÁNH GIÁ

A. Kết quả thực nghiệm

Để đánh giá hiệu quả thuật toán, chúng tôi tiến hành phân tích với 70 mẫu ứng dụng lành tính và 70 mẫu mã độc. Với 70 mẫu ứng dụng lành tính được lấy về từ Google Play [19] và 70 mẫu mã độc được lấy từ Virus Share [20]. Cả 2 mẫu sẽ được chia làm 2 tập: "Tập lấy ngưỡng" và "Tập thử nghiệm". Tập lấy ngưỡng được chúng tôi sử dụng để đánh giá và lấy ra giá trị ngưỡng phù hợp nhất. Dữ liệu thử nghiệm được sử dụng để đánh giá phương pháp mới được đề xuất. Trong thí nghiệm này, các mẫu đầu tiên được phân tích bởi VirusTotal [10], một công cụ quét phần mềm độc hại trực tuyến. Các dữ liệu kiểm tra sự độc hại bao gồm các mẫu không được phát hiện dựa trên phương pháp kiểm tra chữ ký. Dữ liệu học lành tính và dữ liệu thử nghiệm được chọn ngẫu nhiên từ việc thu thập các mẫu ứng dụng lành tính.

Số lượng được phân chia như Bảng 9. Để thực hiện lấy tập tin Android Manifest.xml trong các mẫu mã độc và ứng dụng lành tính nhóm nghiên cứu đã sử dụng công cụ Mobile Security Framework (MobSF). Kết hợp với module nhóm tự phát triển bằng ngôn ngữ lập trình Python để thực hiện bóc tách và đánh giá các mẫu mã độc

cũng như ứng dụng lành tính. Hình 5 là mô hình nhóm xây dựng để thực hiện đánh giá.

BẢNG 8. MỘT SỐ MẪU MÃ ĐỘC SỬ DỤNG TRONG THỰC NGHIỆM

STT	Tên mã độc	Tệp bị nhiễm
1	GinMaster	virusshare_1bdcf187bfc2b0307c5a09fb464e855e virusShare_2a96b4721c638ec5d67b9b318bb0b3e0 virusshare_8669cea30c8178f9d118e89335e4def6 virusshare_2f87990252304e48ebdbd421a466c0a9
2	Lotoor	virusshare_5d5d610fde6e1568926944f12b52d82c virusshare_162cda1ec7f49e9ca2e4c4d47f642041 VirusShare_de873ee0e3652e840ef04b130585e14a virusshare_0fba433339e2f9f5fd846f8d246c95f9 VirusShare_e37d6c0b28c0a94e7b0a94702d5b18da
3	Exploit Linux Lotoor	VirusShare_4e9475c3bd8b240e9979ef1e56194652 virusshare_78ba988e4f67b294cd4bad239a9ac2a2 VirusShare_2941e07bc8596c88b1147547e8fed957 VirusShare_ae6e2a926ce4254db5d9e67769c05d50

Để thực hiện đánh giá các mẫu mã độc và ứng dụng lành tính các mẫu sẽ được chia làm hai phần. Tập lấy ngưỡng và tập thử nghiệm. Tập lấy ngưỡng với mục đích để lấy ra giá trị ngưỡng để xác định giá trị ngưỡng cho 4 tham số được lấy ra trong tệp tin manifest. Sau đó các tệp tin thử nghiệm sẽ được đánh giá dựa trên giá trị ngưỡng được lấy trong tập lấy ngưỡng.

BẢNG 9. BẢNG PHÂN CHIA ỨNG DỤNG

	Tập lấy ngưỡng	Tập thử nghiệm	Tổng số
Ứng dụng lành tính	50	20	70
Mẫu mã độc	50	20	70

Kết quả được thể hiện trong Bảng 10.

BẢNG 10. KẾT QUẢ THỰC NGHIỆM

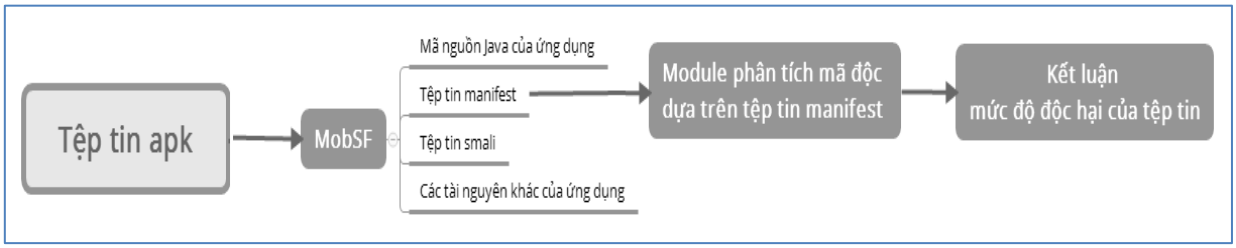
	Tỉ lệ phát hiện đúng (%)	Tỉ lệ phát hiện sai (%)
Ứng dụng lành tính	85	15
Mẫu mã độc	80	20
Trung bình	82.5	17.5

B. Đánh giá

Kết quả thực nghiệm ở phần trước chỉ ra rằng, phương pháp được đề xuất có thể phân loại chính xác các ứng dụng Android. Với dữ liệu càng lớn thì độ chính xác của phương pháp càng cao. Độ chính xác này sẽ phụ thuộc vào giá trị ngưỡng được đề xuất.

Vì là sử dụng phương pháp phân tích tệp tin manifest của ứng dụng Android để phát hiện mã độc nên thời gian phát hiện nhanh hơn rất nhiều so với các phương pháp phát hiện mã độc dựa trên phân tích động. Do không phải chạy ứng dụng trên một môi trường thử nghiệm để kiểm tra các hành vi của ứng dụng, nên phương pháp được đề xuất sử dụng rất ít tài nguyên hệ thống. Đồng thời, để thực hiện phân tích mã độc bằng phương pháp trên cũng không yêu cầu một hệ thống có cấu hình cao. Số lượng tệp tin manifest chỉ có duy nhất đối với tất cả các mẫu và kích thước tệp tin manifest là rất nhỏ so mới kích thươc của tệp tin mã nguồn của ứng dụng nên phương pháp này sẽ có thời gian phát hiện nhanh hơn phương pháp phát hiện bằng chữ ký thông thường.

Tuy vậy, một số mẫu mã độc không bị phát hiện bởi phương pháp này. Phương pháp này vẫn chưa đủ để phát hiện ra các phần mềm quảng cáo. Ngoài hiển thị các thông tin quảng cáo không cần thiết, đồng thời nó chứa một số sự khác biệt nằm ở biên của ứng dụng lành tính và ứng dụng quảng cáo. Điều này gây khó khăn cho phương pháp này trong quá trình phân loại các đặc tính đó.



Hình 5. Mô hình ứng dụng thực hiện đánh giá

VI. KẾT LUẬN

Bài báo này đề xuất một phương pháp phát hiện phần mềm độc hại trên Android. Ưu điểm của phương pháp này là nó chỉ sử dụng các tệp tin manifest để phát hiện phần mềm độc hại. Tệp tin manifest có trong tất cả các ứng dụng Android vì vậy phương pháp này có thể áp dụng cho cả các phần mềm độc hại chưa được biết đến mà không thể phát hiện thông qua phương pháp phát hiện bằng chữ ký thông thường. Hơn nữa, việc chỉ phân tích tệp tin manifest sẽ làm cho chi phí phân tích thấp hơn. Đồng thời, phương pháp này cũng có thể kết hợp với các phương pháp khác để phát hiện chính xác hơn.

Chúng tôi mới chỉ sử dụng một lượng mẫu nhỏ tổng cộng 70 mẫu mã độc và 70 mẫu lành tính. Trong tương lai, chúng tôi sẽ bổ sung và thu thập các mẫu mới để có kết quả chính xác hơn. Đồng thời, chúng tôi đưa vào công cụ Wake [21] với thuật toán Weka J48, dựa trên cây quyết định để lựa chọn ra điểm số ác tính tốt nhất để đưa ra quyết định. Trong phương pháp trích xuất 4 mục thông tin từ tệp tin manifest, các mục thông tin này hoàn toàn có thể dễ dàng thay đổi. Chúng ta nên theo dõi chặt chẽ các xu hướng của phần mềm độc hại trên Android để xác định nên giữ hay sửa đổi kết quả của các mục thông tin nhằm thu được kết quả chính xác nhất.

TÀI LIỆU THAM KHẢO

[1] Christian Lueg, "8,400 new Android malware samples every day", G DATA Security Blog, 2017.

[2] Eric Chin, "Motivations of Recent Android Malware", Symantec Security Response, Tech. Rep, 2011.

[3] Himanshu Shewale, Sameer Patil, Vaibhav Deshmukh and Pragya Singh, "Analysis of Android Vulnerabilities and Modern Exploitation Techniques", in ICTACT Journal on Communication Technology, vol.5, no.1, 2014.

[4] Kindsight, "The Mobile Malware Problem", in A Kindsight White Paper, Ottawa, Canada, Tech.Report, 2012.

[5] Muhammad Zuhair Qadir, Atif Nisar Jilani and Hassan Ullah Sheikh, "Automatic Feature Extraction, Categorization and Detection of Malicious Code in Android Application", in Proceeding International Journal of Information and Network Security, vol.3, no.1, pp.12-17, 2014.

[6] Stefan Brahler, "Analysis of the Android Architecture", Karlsruhe Institute of Technology, Tech. Rep, 2010.

[7] Justin Sahs and Latifur Khan, "A Machine Learning Approach to Android Malware Detection", in Intelligence and Security Informatics Conference, Odense, European, 2012.

[8] Luoshi Zhang, Yan Niu, Xiao Wu, Zhaoguo Wang and Yibo Xue, "A3: Automatic Analysis of Android Malware", in International Workshop on Cloud Computing and Information Security, 2013.

[9] Alessandro Armando, Alessio Merlo and Luca Verderama, "Security Issues in the Android cross-layer architecture", 2012.

[10] Kevin Allix, Tegawende Bissyande, Quentin Jerome, Jacques Klein and Radu State, "Large-Scale Machine Learning-based Malware Detection: Confronting the "10-Fold Cross Validation" Scheme with Reality", in Conference on Data and Application Security and Privacy, San Antonio, Texas, USA, 2014.

[11] Zami Aung and Win Zaw, "Permission-Based Android Malware Detection", in International Journal of Scientific & Technology Research, vol.2, no.3, 2013.

[12] Yousra Aafer, Wenliang Du and Heng Yin, "DroidAPIMiner: Mining API-Level Features for Robust Malware Detection in Android", in Security and Privacy in Communication Networks, pp. 86-103, 2013.

[13] Detecting Android Malware by Analyzing Manifest Files: Ryo Sato1, Daiki Chiba and Shigeki Goto.

[14] Nguyễn Minh Đức, "Phân tích mã độc trên Android và dự đoán xu hướng năm 2015", SecurityDaily, 2015.

[15] Troy Vennon, GTC Research Engineer, "A Study of Known and PotentialMalware Threats", 2010.

[16] Isohara T.; Kawabata H.; Yakemori K.; Kubota A.; Kani J.; Agematsu H.; Nishigaki A. Detection Technique of Android Malware with Second Application. Proceedings of Computer Security Symposium.

[17] Enck W.; Ongtang M.; McDaniel P. On Lightweight Mobile Phone Application Certification.

[18] Wu D.; Mao C.; Wei T.; Lee H.; Wu K. DroidMat: Android Malware Detection throug Manifest and API Calls Tracing. Seventh Asia Joint Conference on Information Security.

[19] https://play.google.com/store

[20] https://virusshare.com

[21] http://www.cs.waikato.ac.nz/ml/weka/

SƠ LƯỢC VỀ TÁC GIẢ

ThS. Lê Bá Cường

Đơn vị công tác: Khoa Công nghệ thông tin – Học viện Kỹ thuật mật mã – Ban Cơ yếu Chính phủ

Email: cuonglb304@gmail.com

Quá trình đào tạo: Nhận bằng cử nhân tại ĐH Khoa học Tự nhiên, ĐH QGHN năm 2007; nhận bằng Thạc sĩ CNTT tại Đại học Công nghệ, ĐH QGHN năm 2011; Hiện đang làm nghiên cứu sinh tại ĐH Công nghệ, ĐH QGHN.

Hướng nghiên cứu hiện nay: Phân tích chương trình phục vụ kiểm thử hồi quy, phát hiện lỗ hổng an ninh, đánh giá sự ảnh hưởng khi thay đổi; Kiểm thử tự động, kiểm thử dựa trên mô hình; Đảm bảo an ninh, an toàn các ứng dụng mobile.

KS. Trịnh Doãn Mạnh

Đơn vị công tác: Công ty cổ phần Công nghệ và Truyền thông Alvasky.

Email: cloudi@alvasky.com

Quá trình đào tạo: Nhận bằng kỹ sư An toàn thông tin tại Học Viện Kỹ thuật mật mã năm 2017.

Hướng nghiên cứu hiện nay: Giải pháp bảo mật cho ứng dụng Android, mã độc trên Android, Giải pháp đảm bảo an toàn cho ứng dụng Web.